Supercharge your upgrade with AutoUpgrade 2.0

Photo by RKTKN on Unsplash

Daniel Overby Hansen

Senior Principal Product Manager

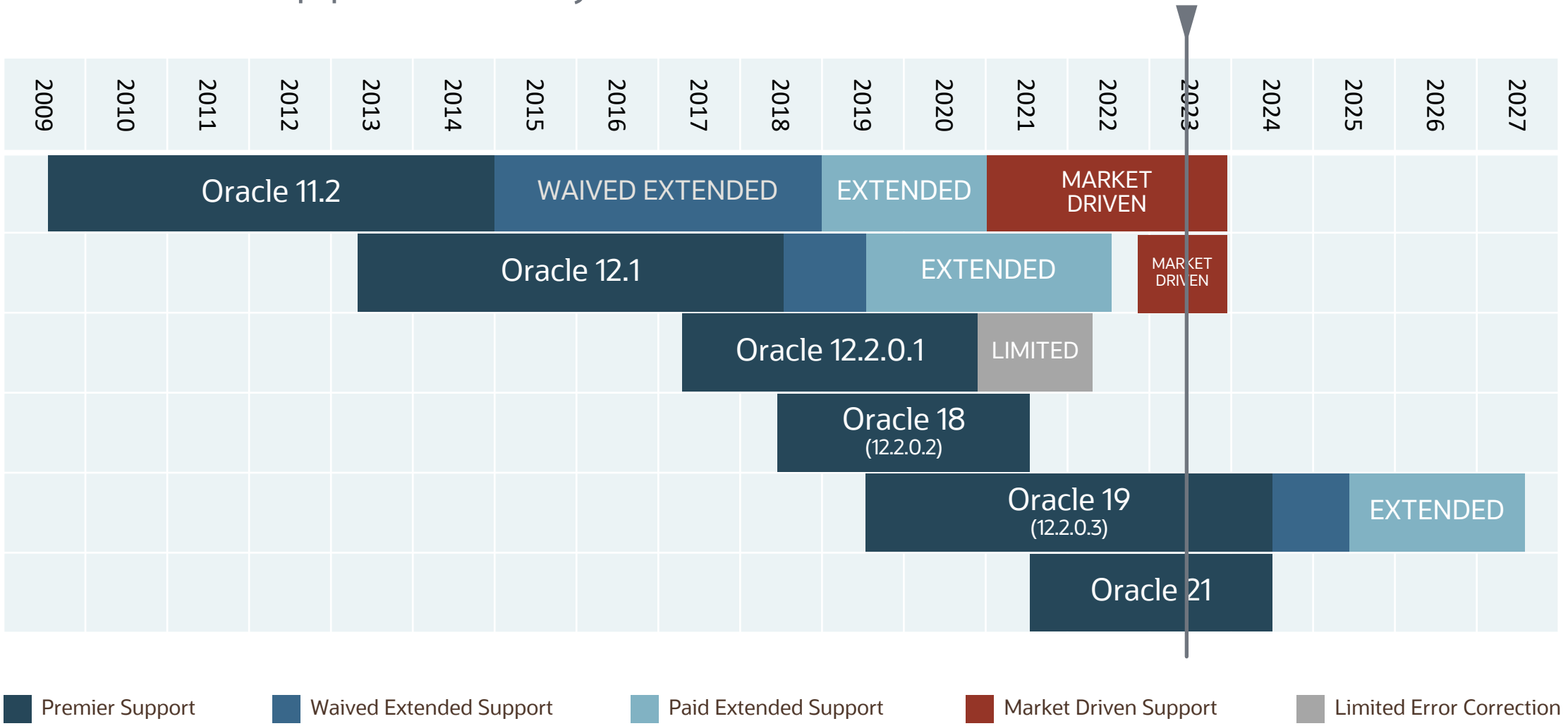
Cloud Migration

dohdatabase

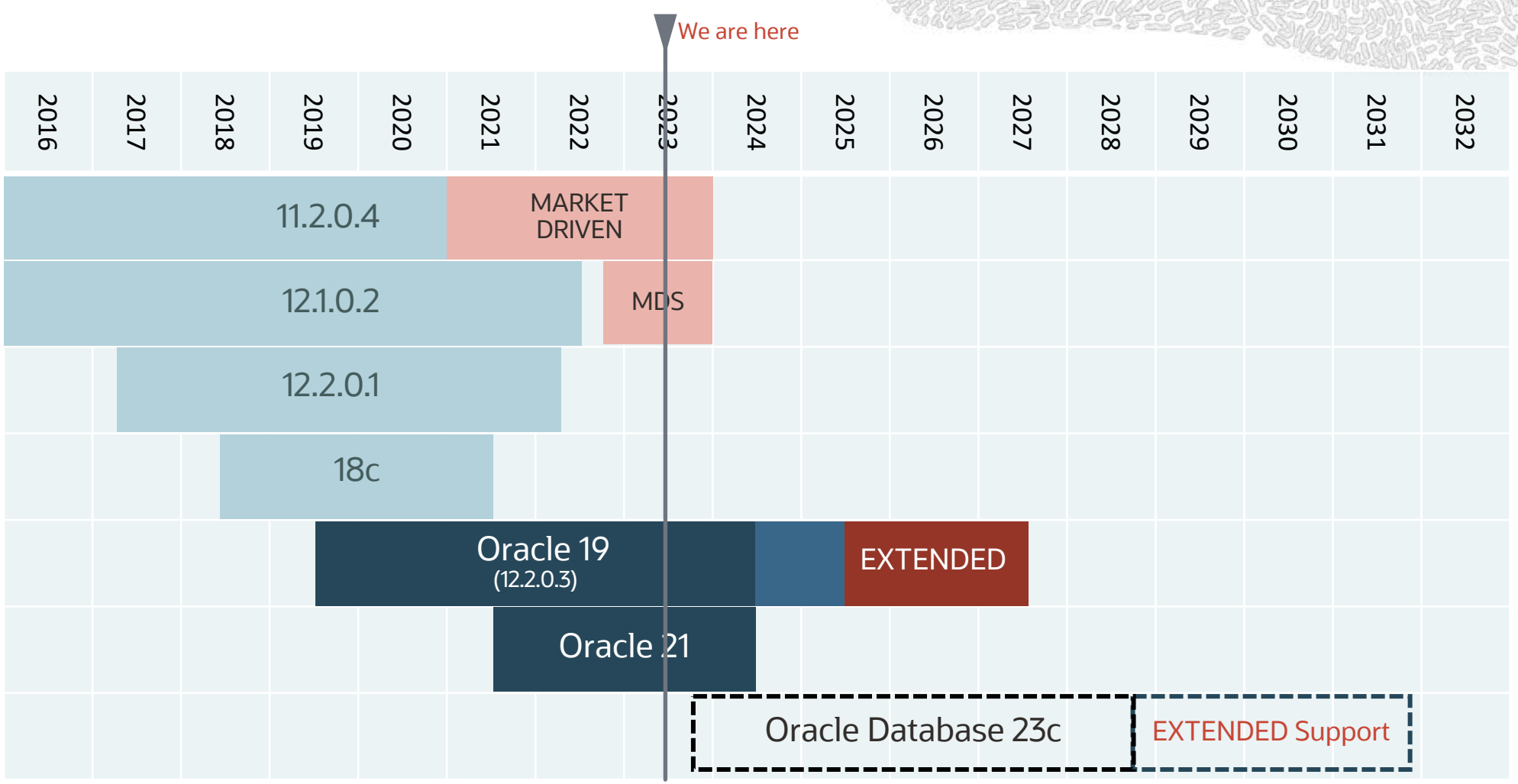
@dohdatabase

https://dohdatabase.com

Lifetime Support Policy



Plan YOUR Release Strategy



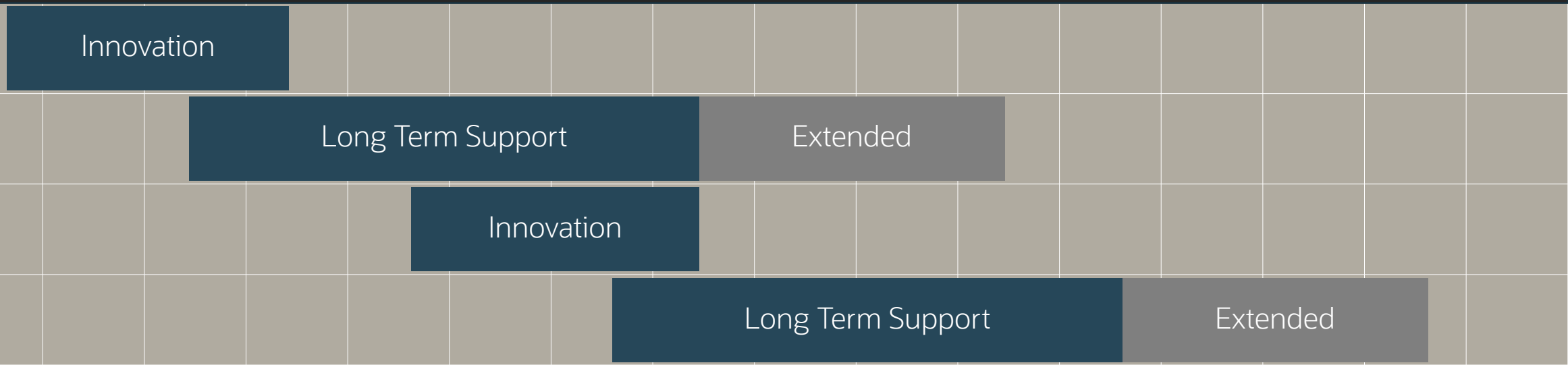
Release Types

LONG TERM SUPPORT

5+ years of Premier Support followed by
3+ years of Extended Support

INNOVATION

2 years of Premier Support
No Extended Support



Move production databases from one
Long Term Support release to the next

Next Long Term Support release

Oracle Database 23c

Upgrade possible only from:

- Oracle Database 19c
- Oracle Database 21c

Do you want to upgrade?

Oracle Database 11.2.0.4
Oracle Database 12.1.0.2
Oracle Database 12.2.0.1
Oracle Database 18c

Oracle Database 11.2.0.4
Oracle Database 12.1.0.2
Oracle Database 12.2.0.1
Oracle Database 18c

Oracle Database 19c

Oracle Database 23c

Everybody must **upgrade to Oracle Database 19c**, with or without Multitenant

AutoUpgrade | Overview

ONE BUTTON APPROACH

But giving you all the control you want

Always use the latest version of AutoUpgrade

Download from My Oracle Support (2485457.1)

```
$ java -jar autoupgrade.jar -version

build.version 23.1.230224
build.date 2023/02/24 14:53:24 -0500
build.hash a1e2990e
build.hash_date 2023/02/24 14:44:39 -0500
build.supported_target_versions 12.2,18,19,21
build.type production
```


Agenda

1

Patching

Database patching
Simple
Easy
Familiar

2

Refreshable Clone

3

Encryption

4

Distributed Upgrade

5

Usability

We made upgrading easy.
Now we make patching just as easy.

AutoUpgrade functionality extended to patching

Patching

1

Install Oracle Home including Release Update, MRP and additional patches (MOS Doc ID 555.1)

2

Create a simple configuration file

3

Start AutoUpgrade in deploy mode

```
$ cat DB19.cfg

patch1.source_home=/u01/app/oracle/product/19.0.0.0/dbhome_19_18_0
patch1.target_home=/u01/app/oracle/product/19.0.0.0/dbhome_19_19_0
patch1.sid=DB19


$ java -jar autoupgrade.jar -config DB19.cfg -mode deploy
```


Patching

USE

Familiar interface

Console

Logging

ANALYZE

Prechecks

Datapatch Checks

Summary report

PROTECT

Resumable

Restoration

Restore point

Fallback

AUTOMATE

`srvctl`

`/etc/oratab`

Files

Datapatch

Patching

Encryption

Hot clone

Refreshable clone

RAC

Proactive fixups

Distributed upgrade

...

Patching

What's missing

Windows

RAC rolling

Data Guard standby-first

Patching

AutoUpgrade

Automate your patching process and benefit from the familiar AutoUpgrade

Fleet Patching and Provisioning

Go fleet scale with FPP and benefit from additional functionality like deployment of Oracle Home

Agenda

1

Patching

2

Refreshable Clone

Unplug-plug upgrades
Non-CDB to PDB
Minimal downtime

3

Encryption

4

Distributed Upgrade

5

Usability

Non-CDB to PDB conversion is irreversible

What are your fallback options?

FALLBACK

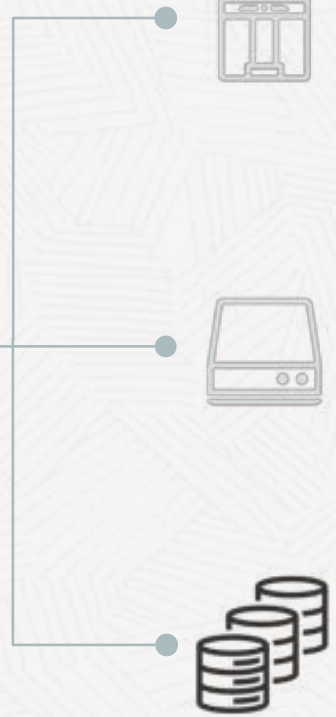
Backup / restore

Ensure you have a recent backup and requires time to restore and recover

Copy data files

Requires time and disk space to hold a copy of the data files

FALLBACK



Backup / restore

Ensure you have a recent backup and requires time to restore and recover

Copy data files

Requires time and disk space to hold a copy of the data files

Refreshable clone

Requires ~~time and~~ disk space to hold a copy of the data files

Requires Oracle Database 12.2 or newer

Refreshable Clone

CREATE

Create PDB from non-CDB over a database link

REFRESH

Apply redo from non-CDB to keep PDB up-to-date

OUTAGE

Disconnect users and refresh PDB for the last time

CONVERT

To become a proper PDB, it must be converted

Refreshable Clone

Source non-CDB	Target CDB

```
CREATE USER dblinkuser
       IDENTIFIED BY ... ;

GRANT CREATE SESSION,
      CREATE PLUGGABLE DATABASE,
      SELECT_CATALOG_ROLE TO dblinkuser;

GRANT READ ON sys.enc$ TO dblinkuser;
```

```
CREATE DATABASE LINK CLONEPDB
       CONNECT TO dblinkuser
       IDENTIFIED BY ...
       USING 'noncdb-alias';
```

Refreshable Clone

Source non-CDB	Target CDB

```
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.source_dblink.NONCDB1=CLONEPDB
upg1.target_pdb_name.NONCDB1=PDB1
```

Refreshable Clone

Source non-CDB	Target CDB

```
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1
```

Refreshable Clone

Source non-CDB | Target CDB

```
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1
upg1.start_time=22/10/2022 02:00:00
--Specify relative start time
--upg1.start_time=+1h30m
```

Refreshable Clone


```
autoupgrade.jar ... –mode deploy
```

```
upg1.start_time=22/10/2022 02:00:00
```

DEMO

- Upgrade to Oracle Database 19c
- Migrate from non-CDB to PDB
- Using Refreshable Clone PDBs

Watch on YouTube

The source non-CDB stays intact to allow fallback

Works for unplug-plug upgrades as well

Pro Tips and Details

Get there faster and smarter

Multitenant Conversion | How Long Does It Take?

DOWNTIME

Requires downtime.

RUNTIME

Typically 10-30 minutes.
Depends mostly on the number of objects.
Not the physical size of the database.

PROCESS

Only need to run it once.
The process is irreversible.
Rerunnable in case of errors.

Ensure archive logs are available on disk during migration

Cloning

CLONING

AutoUpgrade uses `CREATE PLUGGABLE DATABASE` statement with `PARALLEL` clause which clones the database using multiple parallel processes

PARALLEL

Based on system resources and current utilization the database automatically determines a proper parallel degree

TRANSFER

A new file transfer protocol that can bypass several layers in the database to achieve very high transfer rates

NETWORK

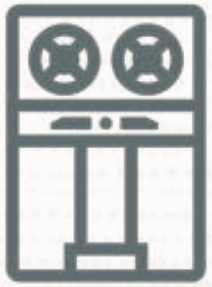
Watch out for network saturation. Optionally, use 3rd party tools like traffic control (`tc`) to limit network usage

```
SQL> select message, sofar, totalwork,time_remaining as remain, elapsed_seconds as ela
     from v$session_longops
     where opname='kpdbfCopyTaskCbk' and sofar != totalwork;
```

MESSAGE	SOFAR	TOTALWORK	REMAIN	ELA
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 643199 out of 1310720 Blocks done	643199	1310720	134	129
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 443007 out of 1310720 Blocks done	443007	1310720	213	109
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 436351 out of 1310720 Blocks done	436351	1310720	216	108
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 370431 out of 1310720 Blocks done	370431	1310720	256	101

```
SQL> select sql_text
     from v$sql s, v$session_longops l
     where s.sql_id=l.sql_id and l.opname='kpdbfCopyTaskCbk';
```

```
SQL_TEXT
/* SQL Analyze(256,0) */ SELECT /*+PARALLEL(4) NO_STATEMENT_QUEUING  */ * FROM X$KXFTASK /*kpdbfParallelCopyOrMove,PDB_FILE_COPY*/
```



Remember a level 0 backup after migration

You can also restore with pre-plugin backups

Agenda

1 Patching

2 Refreshable Clone

3 **Encryption**

Fully supported
Tablespace Encryption
Dedicated keystore

4 Distributed Upgrade

5 Usability

Upgrading and converting
encrypted databases are fully supported

Encryption

Certain database operations require passwords or secrets

```
CREATE PLUGGABLE DATABASE ... KEYSTORE IDENTIFIED BY <password>

ALTER PLUGGABLE DATABASE ... UNPLUG INTO ... ENCRYPT USING <secret>

CREATE PLUGGABLE DATABASE ... DECRYPT USING <secret>

ADMINISTER KEY MANAGEMENT ... KEYSTORE IDENTIFIED BY <password>
```


Secure External Password Store

Operator stores database keystore password
in a Secure External Password Store

AutoUpgrade Keystore

Operator loads database keystore password
into AutoUpgrade keystore ahead of upgrade

Encryption

To configure an AutoUpgrade keystore

```
$ cat DB12.cfg

global.keystore=/etc/oracle/keystores/autoupgrade/DB12
global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=DB12
```


Encryption

Analyze the database for upgrade readiness

```
$ java -jar autoupgrade.jar -config PDB1.cfg -mode analyze
```

Summary report will show which keystore passwords are needed

```
REQUIRED ACTIONS
================
      1.  Perform the specified action ...
      ORACLE_SID                      Action Required
      ------------------------------  -------------------------
      CDB1                            Add TDE password
      CDB2                            Add TDE password
```

DEMO

Upgrading an encrypted database and converting to PDB

Watch on YouTube

Agenda

1

Patching

2

Refreshable Clone

3

Encryption

4

Distributed Upgrade

RAC feature
Performance
PDB availability

5

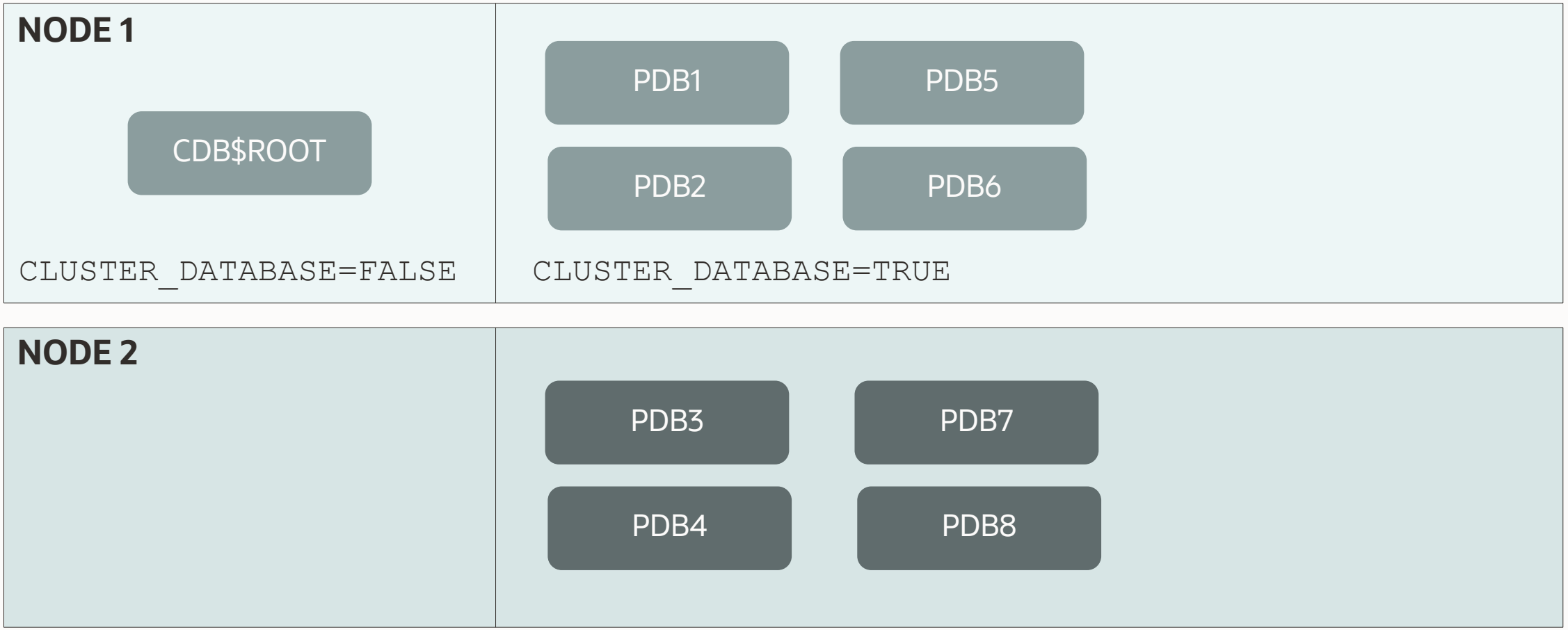
Usability

Distributed upgrade uses all nodes resulting in faster upgrades of CDBs

Applies to RAC and multitenant architecture only

Distributed Upgrade

Distributed Upgrade



Distributed Upgrade

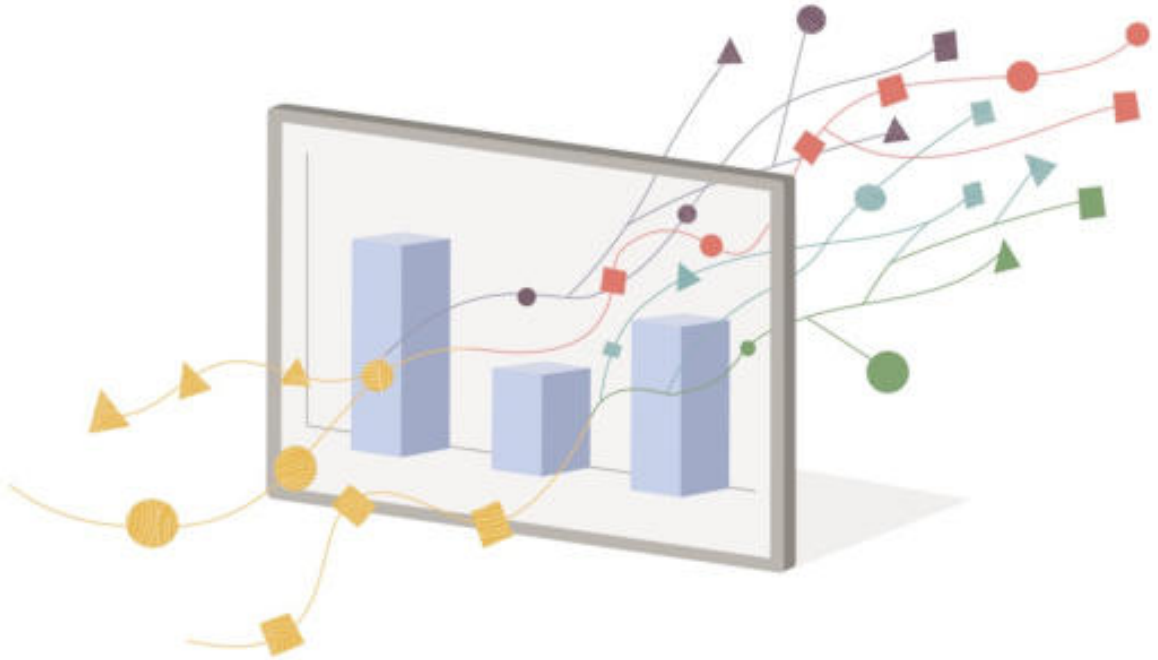
To enable distributed upgrade

```
$ cat RACDB.cfg

global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
upg1.log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade/RACDB
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=RACDB
upg1.tune_setting=distributed_upgrade=true


$ java -jar autoupgrade.jar -config RACDB.cfg -mode deploy
```

41%

In benchmark, time saved
by using distributed upgrade

- 2 node RAC database
- 4 CPUs each
- CDB with 8 PDBs



By default,
AutoUpgrade uses two nodes

Distributed Upgrade

To enable more nodes

```
$ cat RACDB.cfg

global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
upg1.log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade/RACDB
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=RACDB
upg1.tune_setting=distributed_upgrade=true,active_nodes_limit=n


$ java -jar autoupgrade.jar -config RACDB.cfg -mode deploy
```

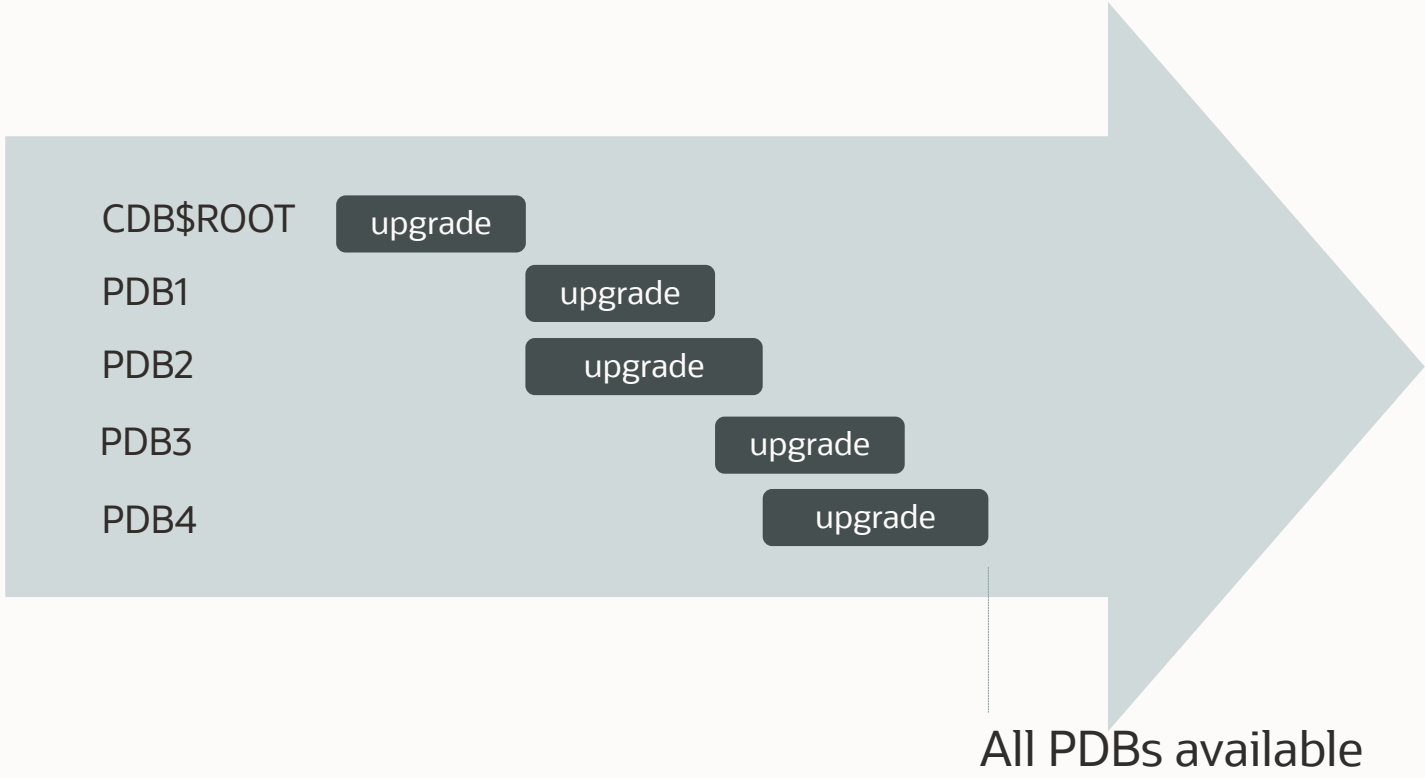

Some PDBs are more important

Control the order of upgrade

PDB Availability

DEFAULT

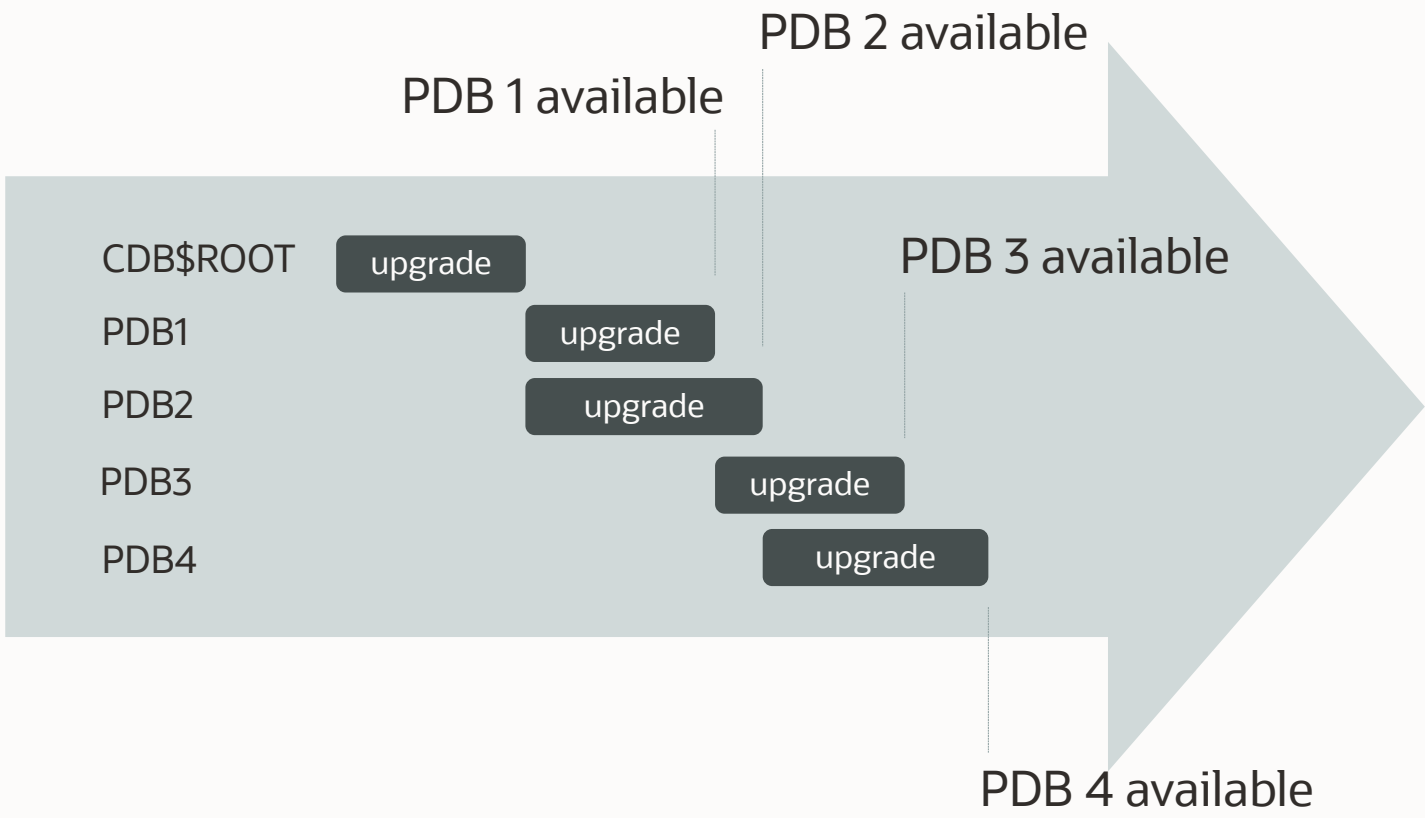
```
make_pdbs_available=false
```



PDB Availability

IMMEDIATELY AVAILABLE

`make_pdbs_available=true`



```
alter pluggable database SALESPROD priority 1;

alter pluggable database SALESDEV priority 2;

alter pluggable database SALESUAT priority 2;

alter pluggable database SALESTEST priority 3;
```

Agenda

1

Patching

2

Refreshable Clone

3

Encryption

4

Distributed Upgrade

5

Usability

Repeating commands
History
Fixup list

DEMO

Enhancing the console experience

Watch on YouTube

```
--Repeat lsj command every 10 seconds
lsj -a 10

--Repeat status command every 10 seconds
status –jobid n –a 10

--Repeat last command
/

--Show history of commands
h

--Repeat command number n from history
/n
```

```
--Show fixups for a job
fxlist –job n

--Disable a fixup
--Example: fxlist -job 100 -c DB12 alter
OLD_TIME_ZONES_EXIST run no
fxlist -job n -c <container> alter <fixup>
run no
```


THANK YOU

Visit our blogs:

https://MikeDietrichDE.com

https://DOHdatabase.com

https://www.dbarj.com.br/en

THANK YOU

Webinars:

https://MikeDietrichDE.com/videos

YouTube channel:

OracleDatabaseUpgradesandMigrations

THANK YOU